Attorney's Docket No.: 10559/350001/ P9864

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: DIGITAL SYSTEM OF ADJUSTING DELAYS ON CIRCUIT BOARDS

APPLICANT: RANDY BONELLA AND JOHN HALBERT

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL558601705US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

September 15, 2000
Date of Deposit

Signature

Derek W. Norwood
Typed or Printed Name of Person Signing Certificate

DIGITAL SYSTEM OF ADJUSTING DELAYS ON CIRCUIT BOARDS

BACKGROUND

Lengths of lines on a computer circuit board may affect the signals passing through the system. For example, system buses often run a number of lines in parallel. The lengths of the lines, however, may cause a delay between the times when the signals arrive.

In order to minimize the length induced delay between the different elements of the bus, layout considerations have been used. For example, trial and error techniques may be used to fine tune the signal paths, to allow signals to arrive in synchronism.

Extra lengths, such as serpentines, may be added at different areas on the layout.

BRIEF DESCRIPTION OF THE DRAWINGS

The drawings show:

Figure 1 shows a hardware implementation for levelization;

Figure 2 shows a flowchart of levelization; and

Figure 3 shows a flowchart of clock skew operation.

DETAILED DESCRIPTION

The present invention teaches a system which addresses levelization. This system finds application in, for example, computer systems, such as memory interconnects, front side bus interconnect, graphics devices, I/O interconnects, and any other device which uses multiple bit interconnections.

A hardware solution is described which produces specified types of programmable delays in such a system. The hardware solution can include a register which sets the desired delay on each of the plurality of lines.

A first embodiment is shown in Figure 1. An element 100 produces a number of outputs shown as bus 110. The bus 110 can include a plurality of lines, which may or may not have mismatch routing delays. These lines are buffered by I/O buffers 115. Each of the lines 120, 122, 124, 126 is shown in Figure 1. While only four lines are shown in Figure 1, it should be understood that the bus may have many more lines. For example, buses often have 8, 16 or 32 or more lines.

Programmable delay elements 130, 132, 134, 136 are provided on the lines. In this embodiment, there is a single delay element for each line, however, one or more of the lines may be configured without the delay element. Each delay element is controlled by a respective control line 138. In addition, there can be multiple delay lines for each line.

The signals 120, 122, 124, 126 are delayed by the respective delay elements to form (delayed) output signals 160, 162, 164, 166. These signals are coupled to the core logic 170, which can be circuitry that depends on the function of the device, e.g, memory, graphics, motherboard chipsets or other applications described herein.

A levelization register 140 stores a plurality of values. It can store a single different value for each of the delay elements 130 - 136. These delay values can delay the signals by specified amounts; e.g., by amounts which cause the signals to arrive substantially simultaneously, or within predetermined times of one another.

Arbitration logic 150 carries out the determination of values to be stored in the levelization register. Arbitration logic 150 can be coupled to the output lines 160, 162, 164, 166. The arbitration logic 150 reviews the signals 160 through 166, and determines desired timing information. For example, the arbitration logic 150 may determine whether adjacent lines are leading or trailing each other. Based on this determination, the settings for delay elements 130-136 are determined. These settings cause the values on output lines 160, 162, 164, 166 to be delayed to a specified relationship, e.g., substantially synchronized, or specifically offset.

The information is stored in this levelization register 140. The values are used to delay, bidirectionally, the signals on the signal path.

Alternatively, the arbitration logic 150 can determine leading and lagging edges of signals, and can program those values in the levelization register.

The programmable delay elements 130-136 can be formed using any element which can delay a signal by a variable amount based on an applied signal. These can include a digital delay element, a phase locked loop (PLL) which has multiple taps, a digital locked loop (DLL), or other techniques such as current starving transistors to delay the propagation of the signal therethrough. Moreover, as described above, not each line needs a delay element, e.g., some of the output lines may not include delay elements. However, it may be preferred that each of those lines which includes the delay element includes its own separate delay line 138.

This system can be used in a number of different applications. One application of the present system is in high speed parallel interconnection, where the bus length may actually affect the way that the data arrives at the core logic. This can also be used in memory interconnects, front

side bus interconnect, graphics, IO interconnects and the like.

As described above, the arbitration technique carried out by arbitration logic 150 determines the values that are stored in the levelization register 140, which in turn stores contents for leveling mismatch in the delay elements 130 -136.

Different ways of carrying out the arbitration are disclosed herein.

In general, the arbitration in arbitration logic 150 need only be carried when there is a system event. A system event could include, for example, a change of components installed in the system, such as new memory and/or new cards installed or anything else that might change some issue associated with signal delay. Different kinds of events that should cause new arbitration can be defined. In a Windows ™ system, the events can be stored in the system registry and can be triggered by the Plug and Play™ detecting system, for example.

A catastrophic system problem such as a crash, or the like can also cause a new arbitration to be carried out.

The flowchart in Figure 2 shows the arbitration technique. It can be run by a processor, a microcontroller, or can be executed using dedicated hardwired logic, programmed into a field programmable gate array or into discrete circuitry using Hardware Design Language.

The system operates starting in a hard boot at 200 where a special flag is determined at 205. The flag indicates whether a system event of the defined type has occurred. If not, control passes to 210, where levelization values are restored from non volatile memory 142. The information is stored in the levelization register 140, and subsequently used for levelization of signals at 215.

If a system event has occurred at 205, then the system runs through a process of levelizing skews of the interconnects, beginning at 220. Interconnect skew levelization first comprises the master issuing a signal to the "slave" device 100, to cause signals to be produced on the external data bus 110. Each line is dithered by adding a certain amount of delay amount. When all signals arrive at the same time, or within a specified resolution of one another, then the current delays in the levelization register are taken as being the proper levelization delays. This is carried out physically by having checking the alignment of edges at 230. If the edges are not aligned at 235, the arbiter issues a change to levelization register to check the alignment using the next dither value. Flow returns to 225 where the master issues a signal to the slave to again produce signals to produce a new test.

When the edges are aligned at 230, then the current settings are taken as being proper, and the levelization register settings are read and stored in the non volatile memory 142.

A special margin test operation is shown as line 250. The margin test operation in more detail in Figure 3. System timing margin is well-known in the art as being the level that is needed to set up and hold lines to the clock edge. However, in this system, programmable delays can be changed under user control. Because of these programmable delays, certain devices operate outside of the normal timing margins. This means that parts of the system are operating at a time that is delayed relative to the operation of other parts of the system.

System timing margin can be carried out in the system as shown in Figure 3. Again, the delays in the levelization register 140 are dithered at 300. Each time a different dither occurs, the master issues a signal to the slave at 305, and a data comparator in the core logic 170 checks for failed data. If the system passes, then those values are taken as being usable, and the next value is used. Any failed values are read at 315, compared to the expected margin at 320, and used to form a report. The report 325 is used to set values for the levelization register. These values do not

necessarily cause the signals to arrive simultaneously, but rather cause the signals to arrive in a way that allows some separated objects in the core logic to operate relative to one another, as desired. For example, certain parts may be delayed intentionally relative to other parts for some system timing reason.

Although only a few embodiments have been disclosed in detail above, other modifications are possible. For example, other programmable delay elements could be used.